ROI creation on HLT

Giulia Casarosa & Eugenio Paoloni

INFN - Sezione di Pisa

23rd October 2013

Outline

- ROI basf2 Modules on HLT nodes and HLT output nodes
- The PXD Data Reduction Module Algorithm
- Performance of the Module
  - efficiency, data reduction factor, execution time
    - for the Belle2 and the test-beam configurations
- Conclusions

ROIs in the DAQ picture

Onsen

TCP IP

HLT out

basf2 Module:
1. ROISender

Event n
SVD hits

Event n +...
SVD hits

Event n + ...
SVD hits

basf2 Modules (on each node):
1. **PXDDataReduction**
2. ROIPayloadAssembler

ROI of Event n

ROI of Event n+...

ROI of Event n+.....

PXDDataReduction Module is the one that compute the ROIs

ROIPayloadAssembler Module pack the ROIs in the format ready for the ONSEN

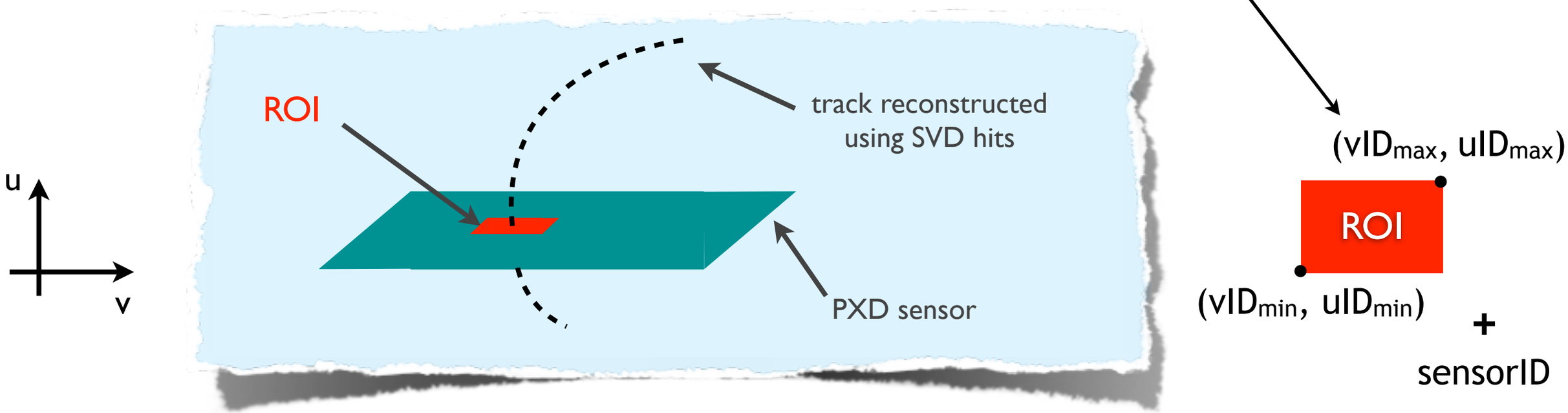
Event builder #1

HLT nodes
running basf2 executables

Software-Based Data Reduction Algorithm

[implemented in the PXDDataReduction]

1. pattern recognition performed with SVD hits only:
   - ★ GFTrackCand list produced by *VXDTF (*or *MCTrackFinder* for testing purposes only*)*
2. fit the GFTrackCand using the standard kalman filter (genfit) and produce a GFTrack
   - ★ the fit is done in both directions: first inward, then outward
3. the GFTrack is extrapolated on each of the 40 planes containing a PXD sensor
   - ★ obtain an extrapolation point on the plane and the associated statistical errors σ_{stat}
4. a rectangular region is defined given σ_{stat}, a systematic error σ_{syst} and a total number of $\sigma = \mathrm{sqrt}(\sigma^2_{stat} + \sigma^2_{syst})$ in each direction u,v
5. the region is intersected with the sensor and then translated in pixels ID



ROI Efficiency with MCTrackFinder

➡ Definition of efficiency for PXD Data Reduction:

$$\varepsilon = \frac{\text{\# PXDDigits inside a ROI}}{\text{total \# PXDDigits of GFTrackCand}}$$

inefficiencies of the pattern recognition are factorized!!

➡ We've simulated 1k events using EvtGen and use the *MCTrackFinder* as pattern recognition:

- ~10 tracks/event
- ~4.2 PXDDigits/track

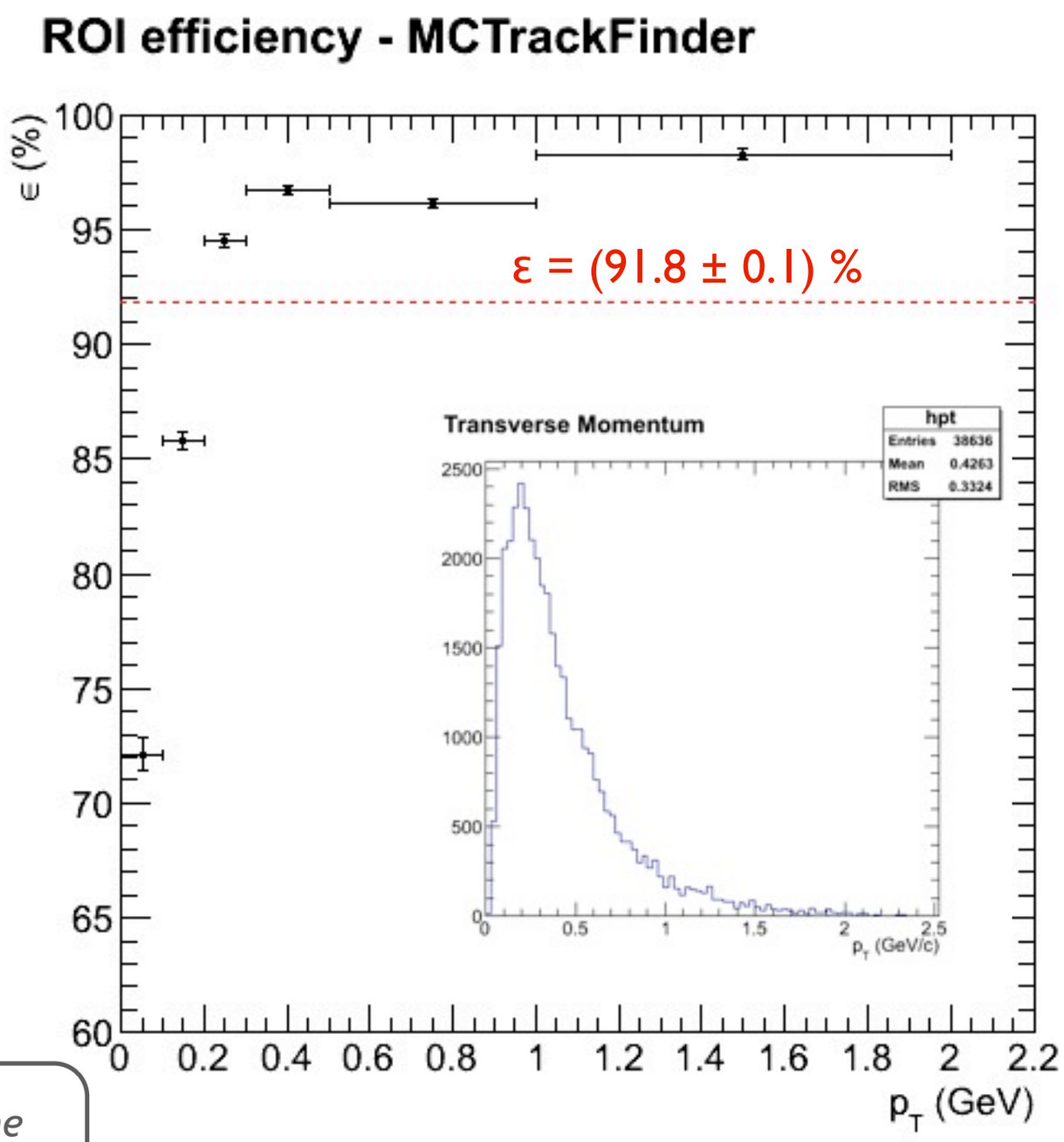
track candidates are built with the true SVT hits

➡ *Efficiency* = (91.8±0.1)% = 38636/42072 PXDDigits

- strongly dependent on the *transverse momentum (see next slide)*

ROI Efficiency, p_T dependence

- ➡ *Efficiency* = (91.8±0.1)% = 38636/42072 PXDDigits
  - strongly dependent on the *transverse momentum*
- ➡ *Inefficiency* mostly due to failures in fitting the track and finding an intercept with the sensor planes
  - 94% of the times no intercept is found
    - increasing the size of ROI will not have a significant impact on the efficiency
  - 6% of the times a ROI is defined, 95% of which the sensor is the wrong one.

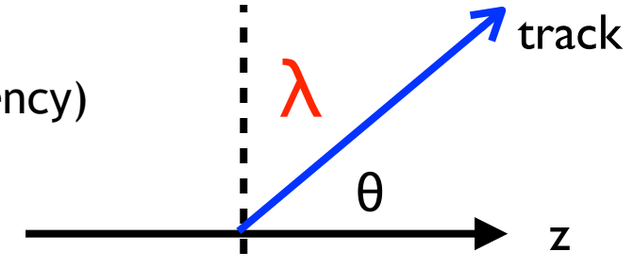


$$\varepsilon = \frac{\#\ \text{PXDDigits inside a ROI}}{\text{total \# PXDDigits of GFTrackCand}}$$

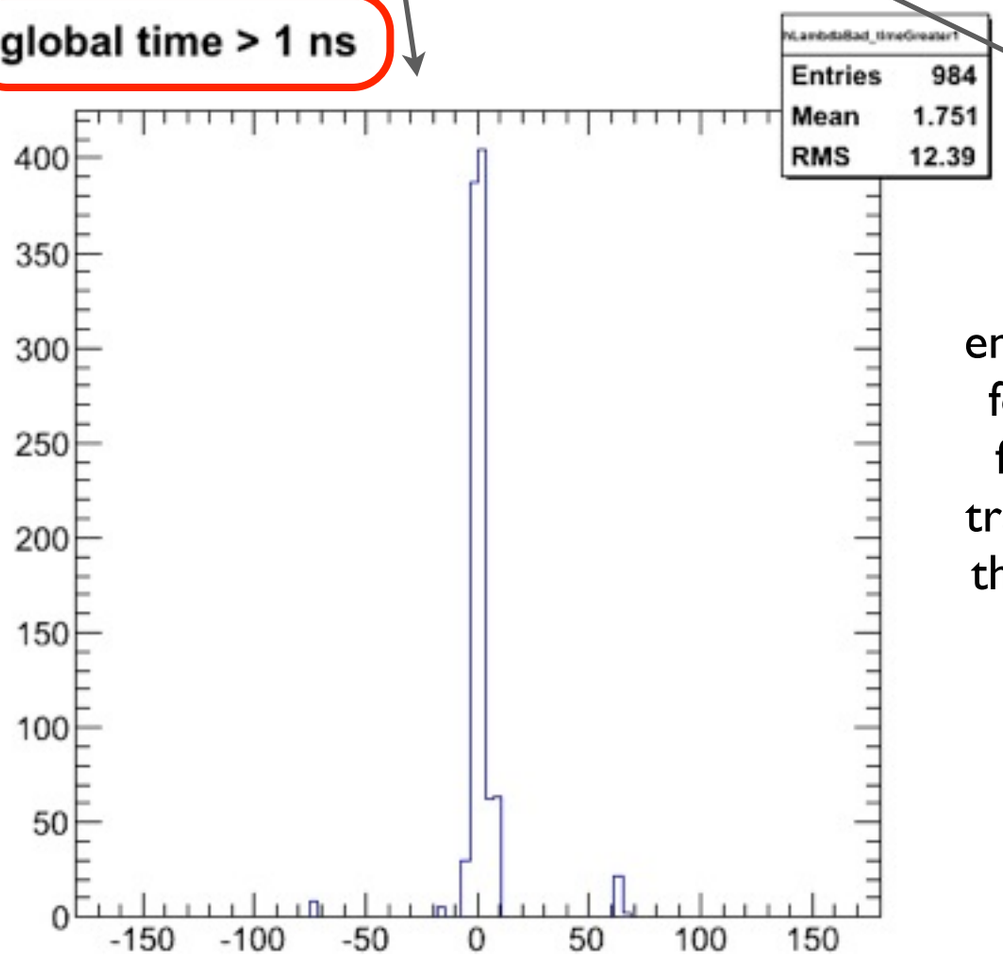
inefficiencies of the pattern recognition are factorized!!

What Hits are we Missing?

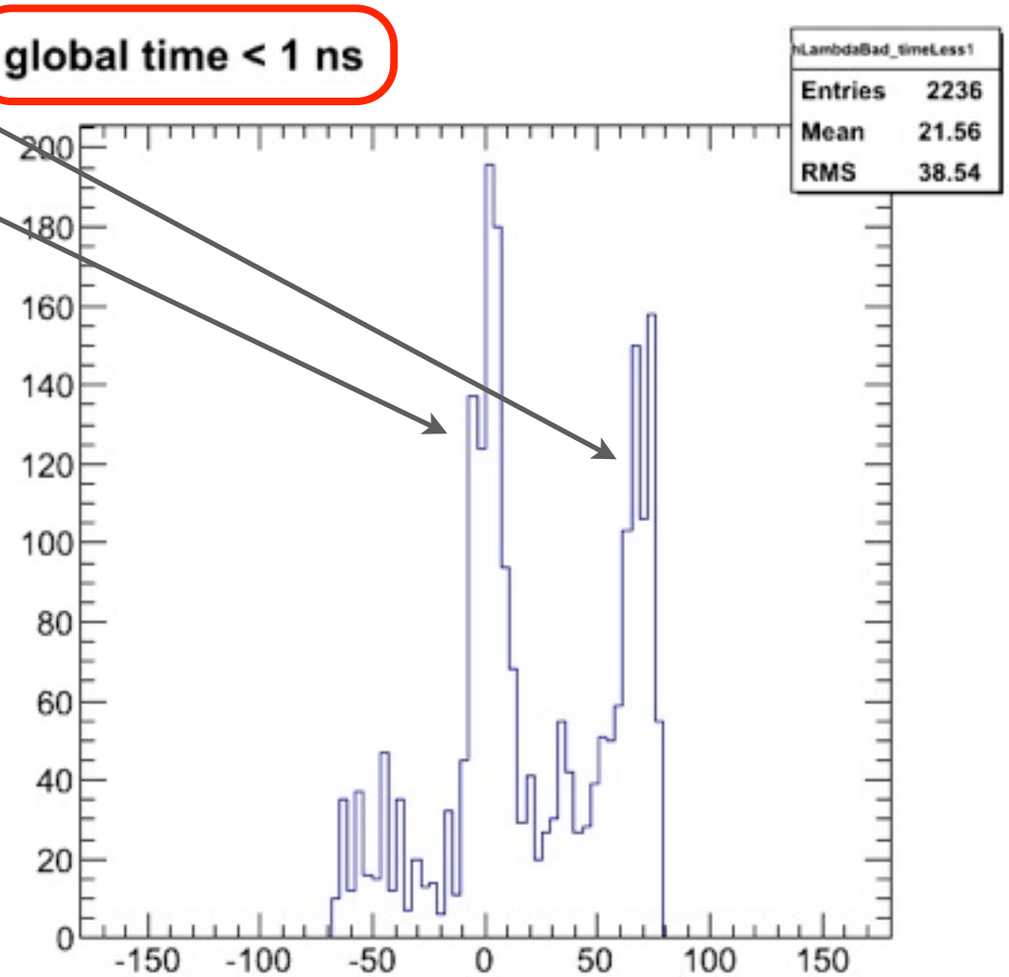
- mainly hits of low transverse-momentum tracks
- later hits of tracks looping on the plane z ≈ 0 (~30% of the inefficiency)
- first hits of tracks at λ≈0° and λ≈65° (~ 70% of the inefficiency)



global time > 1 ns

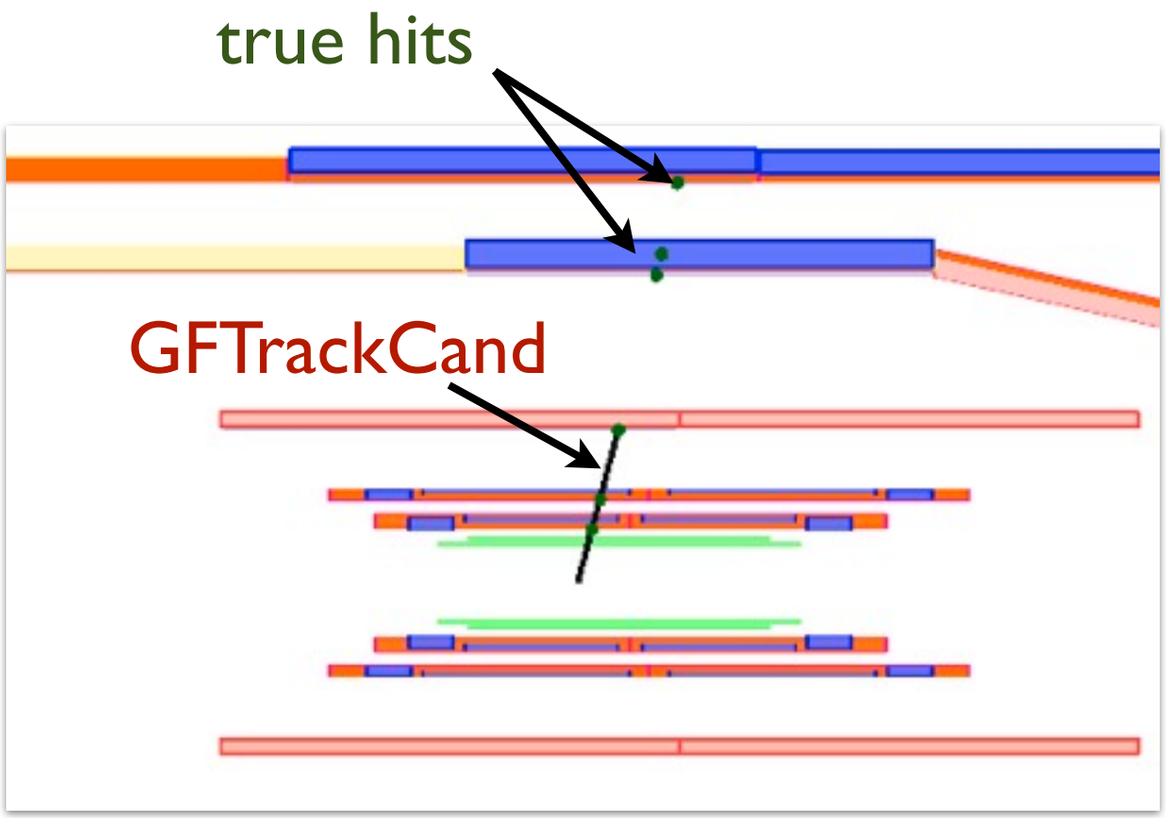


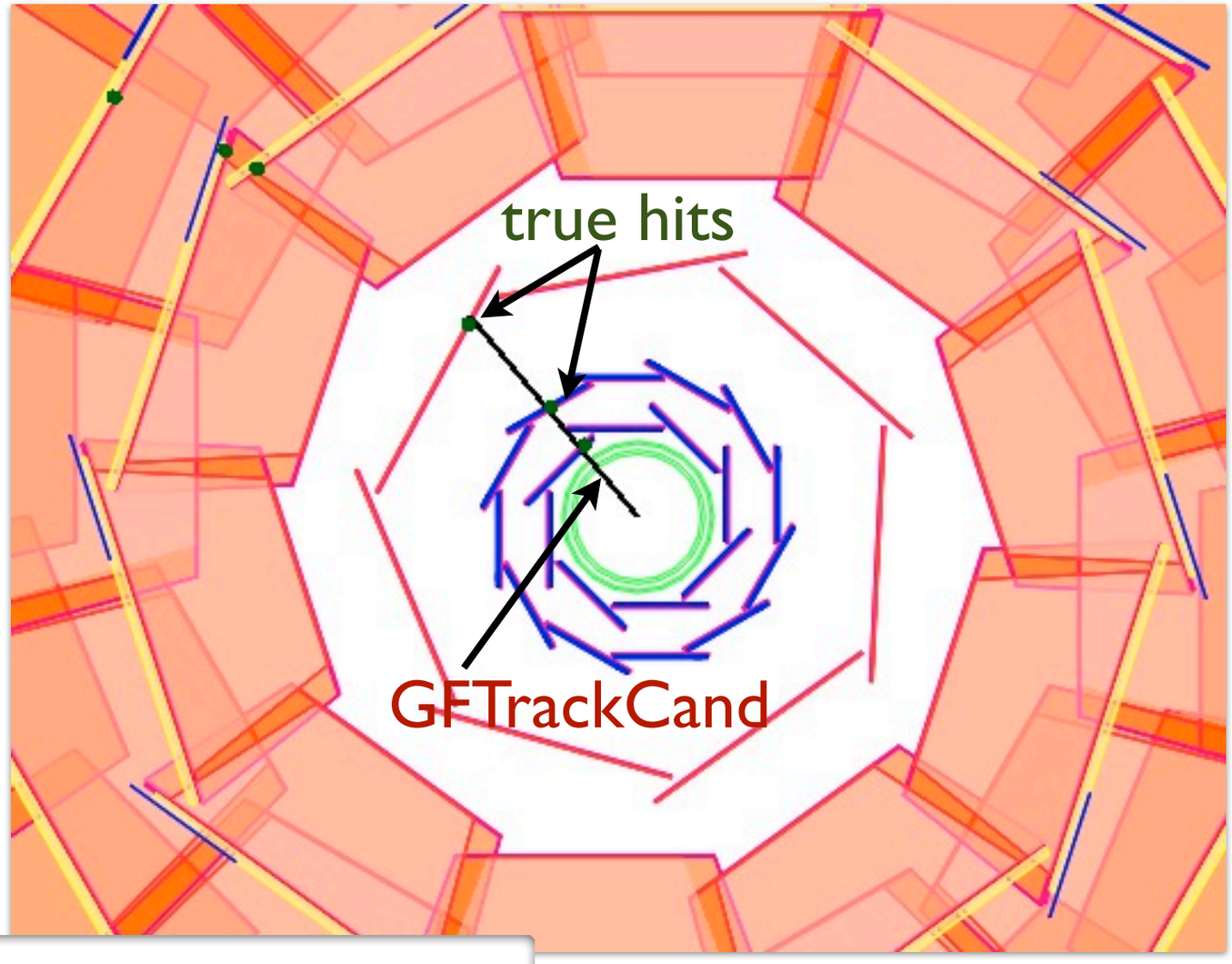
global time < 1 ns



entries = tracks for which the fit has a "bad track status" or the intercept is not found

Inefficiency due to *Bad Track Status*



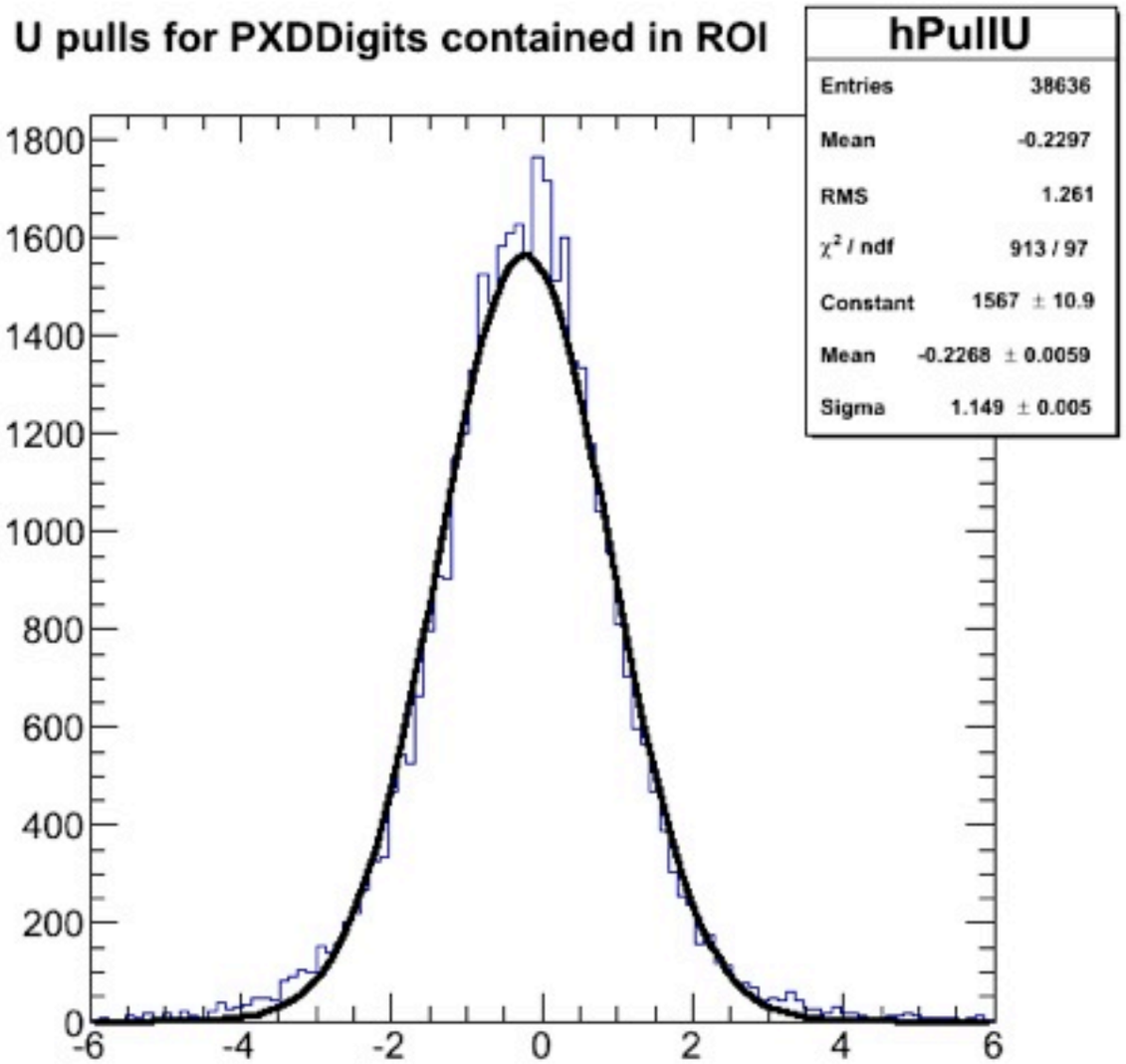


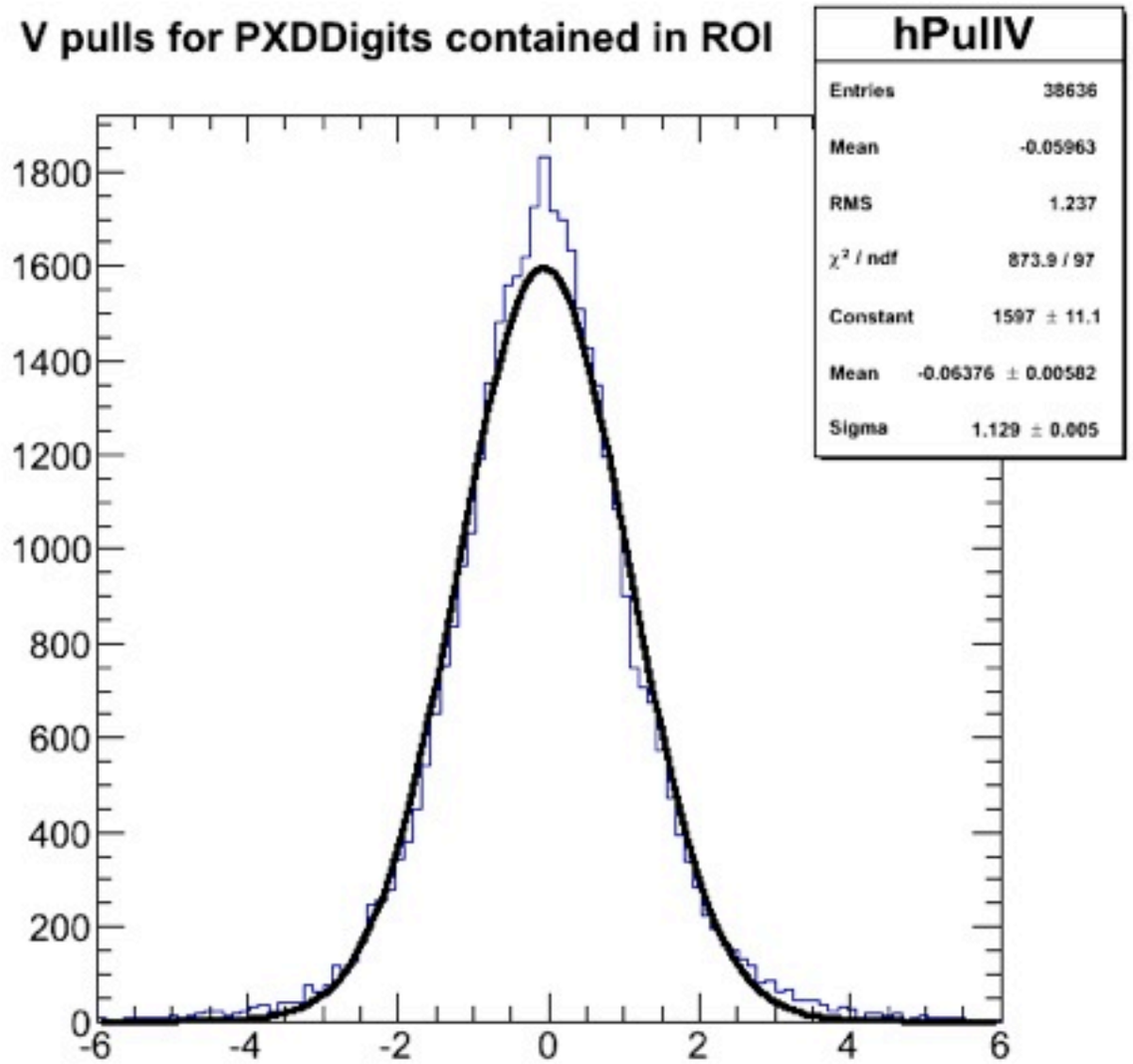

```
GFException thrown with excString:
RKTrackRep::RKutta ==> Do not get closer to plane!
in line: 1230 in file: /home/buildbot/externals/v00-04-01/src/genfit/RKTrackRep/RKTrackRep.cxx
with fa
[WARNIN
```

```
GFException thrown with excString:
RKTrackRep::Extrap ==> maximum number of iterations exceeded
in line: 934 in file: /home/buildbot/externals/v00-04-01/src/genfit/RKTrackRep/RKTrackRep.cxx
with
[WARN
```

```
GFException thrown with excString:
RKTrackRep::RKutta ==> momentum too low: 2.56996 MeV
in line: 1134 in file: /home/buildbot/externals/v00-04-01/src/genfit/RKTrackRep/RKTrackRep.cxx
with fatal flag 0
[WARNING] bad track status  { module: PXDDataReduction }
```

ROI Definition, the U and V pulls





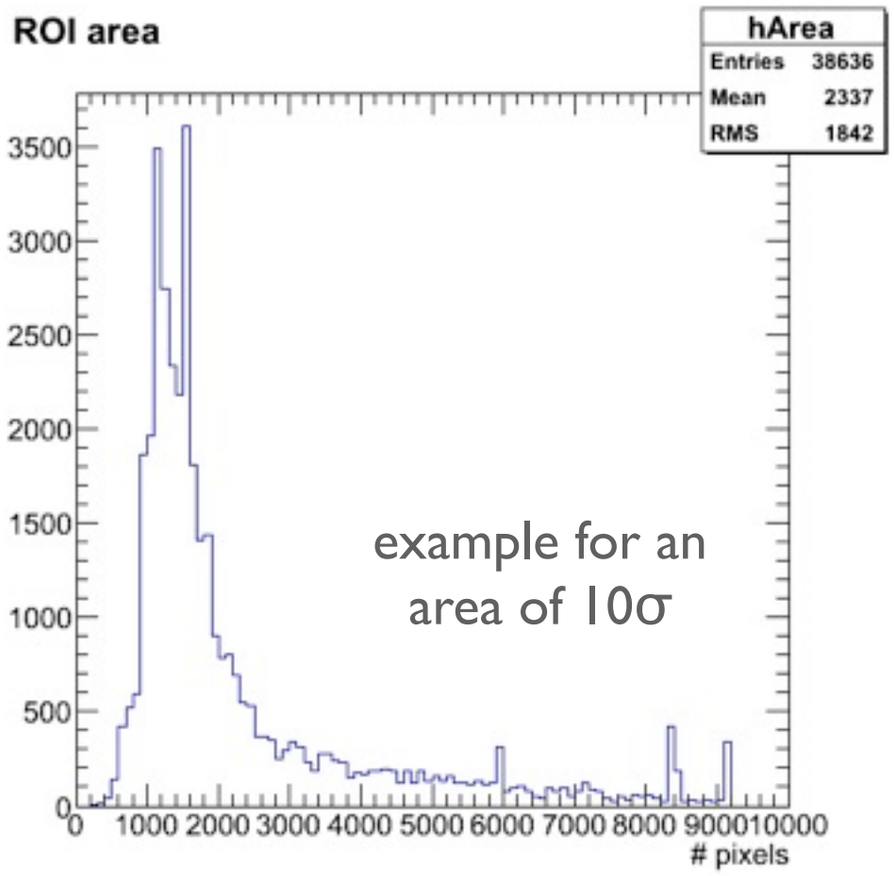
- pull = (intercept – center of the fired pixel)/(stat error on intercept)
- U (V) pull are slightly biased to negative values by 20% (5%) of the stat error
- U, V stat errors are underestimated by ~10-15%

Data Reduction Factor & Execution Time

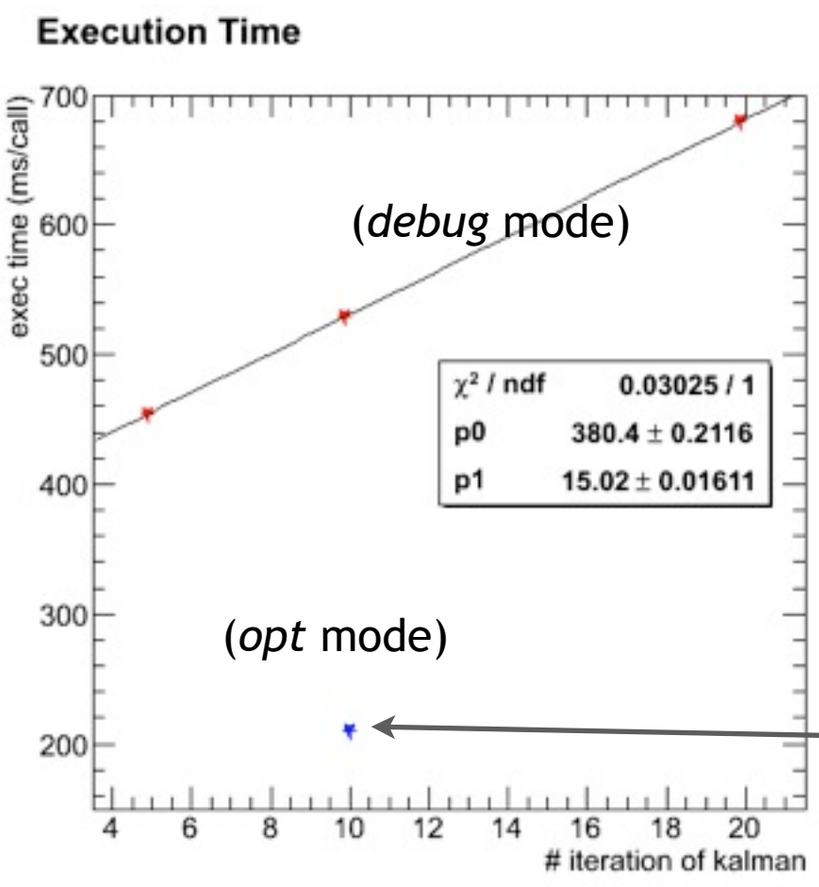
➡ Data Reduction Factor = $\frac{\text{< \# pixels in ROI/event >}}{\text{250*768 pixels/module * 40 modules}}$

- 1.2% taking 10σ in each direction
- 0.4% taking 5σ in each direction

(to be tested with background)



example for an area of 10σ



➡ Execution time (code compiled in *debug* mode):

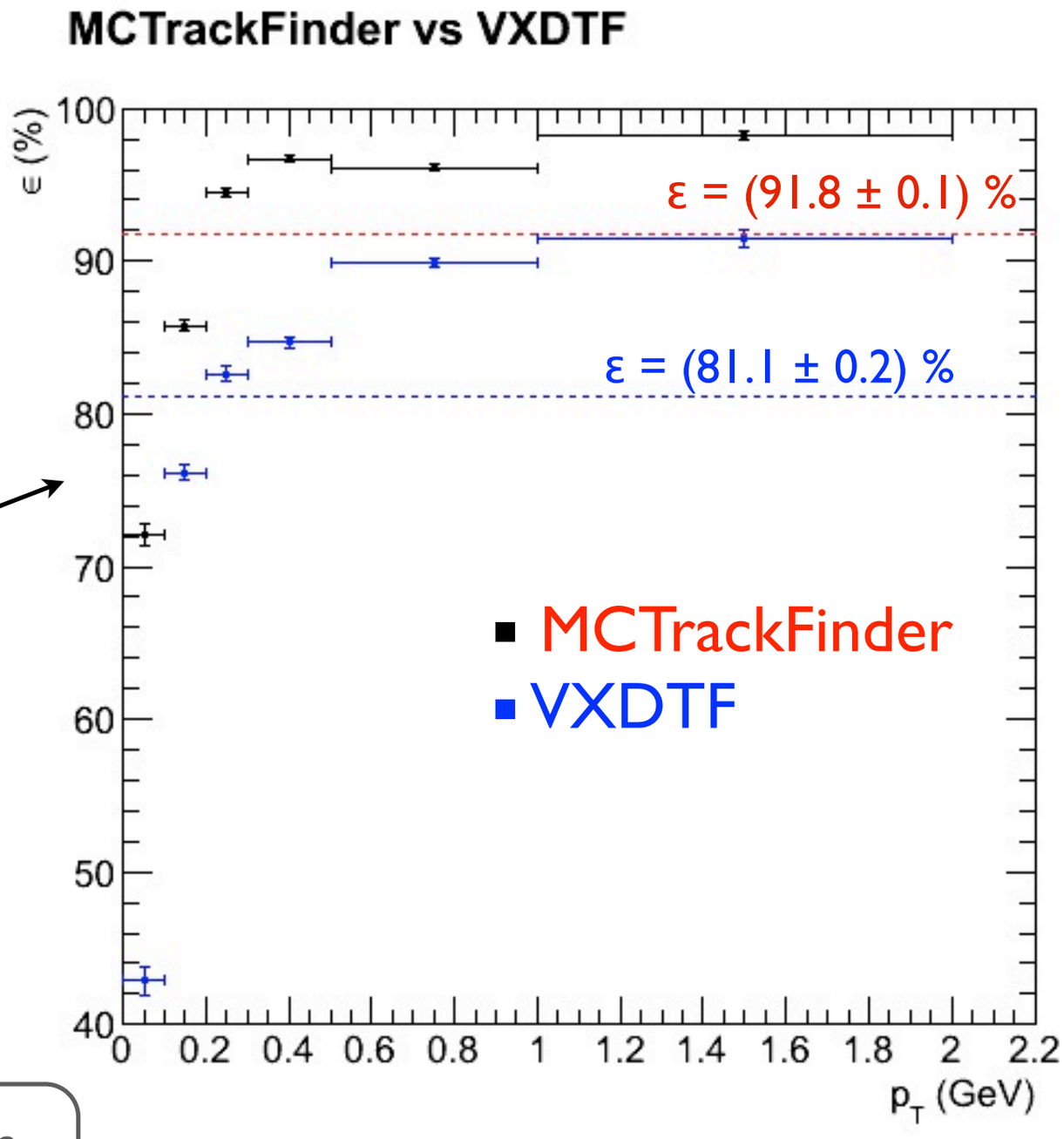
- ★ 45.5 ms/track with 5 iterations of the kalman filter
- ★ 53 ms/track with 10 iterations of the kalman filter
- ★ opt mode: 21 ms/track with 10 iterations
- ★ 68 ms/track with 20 iterations of the kalman filter

MCTrackFinder vs VXDTF

track candidates are built with the true SVT hits

track candidates produced by real pattern recognition

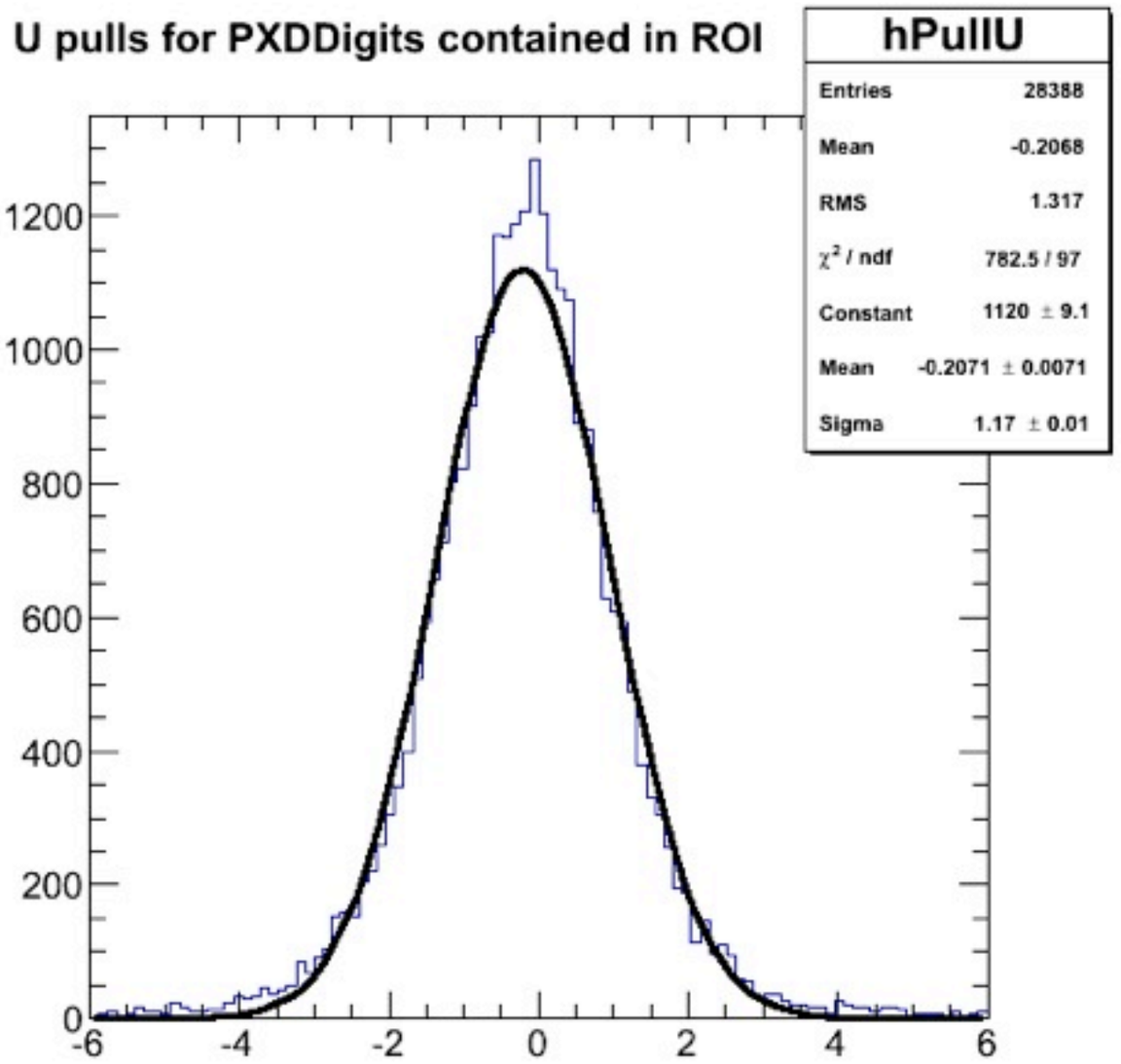
- We simulate 1k events using EvtGen and use the *VXDTF* as *pattern recognition*:
  - ~8.5 tracks/event
  - ~4.1 PXDDigits/track
- Efficiency = (81.1±0.2)% = 28388/34988 PXDDigits
  - similar dependence on transverse momentum observed with MCTrackFinder
- Inefficiency mostly due to failures in fitting the track and finding an intercept with the sensor planes
  - similar behavior observed with MCTrackFinder

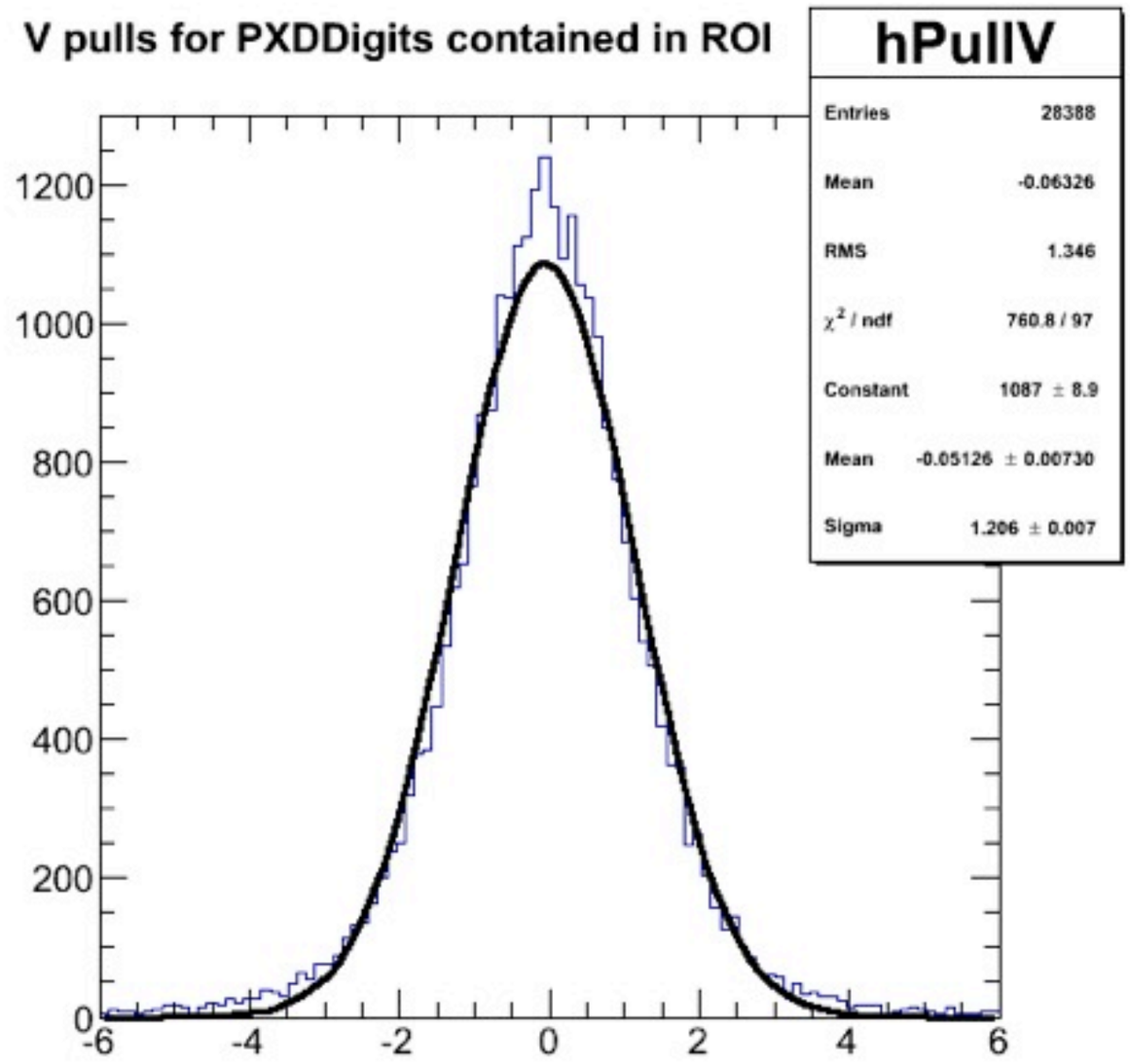


$$\varepsilon = \frac{\text{\# PXDDigits inside a ROI}}{\text{total \# PXDDigits of GFTrackCand}}$$

inefficiencies of the pattern recognition are factorized!!

ROIs with the VXDTF





- U (V) Pulls are negatively biased by 20% (5%) of the statistical error
- the statistical errors are underestimated by ~10-15%
- Data Reduction Factor = 0.8%
- Execution time = 35 ms/track (10 iterations of the kalman filter)

Performance for Beam Test

- We simulate 1k events using particleGun (2GeV e^-, no beam divergence) and use the *VXDTF* as *pattern recognition*:
  - ~0.78 tracks/event (0.83 with MCTrackFinder)
  - ~3.3 PXDDigits/track
- Efficiency = (99.0±0.2)% = 2588/2615 PXDDigits
- Data Reduction Factor = 0.05%
- Execution time = 2 ms/track

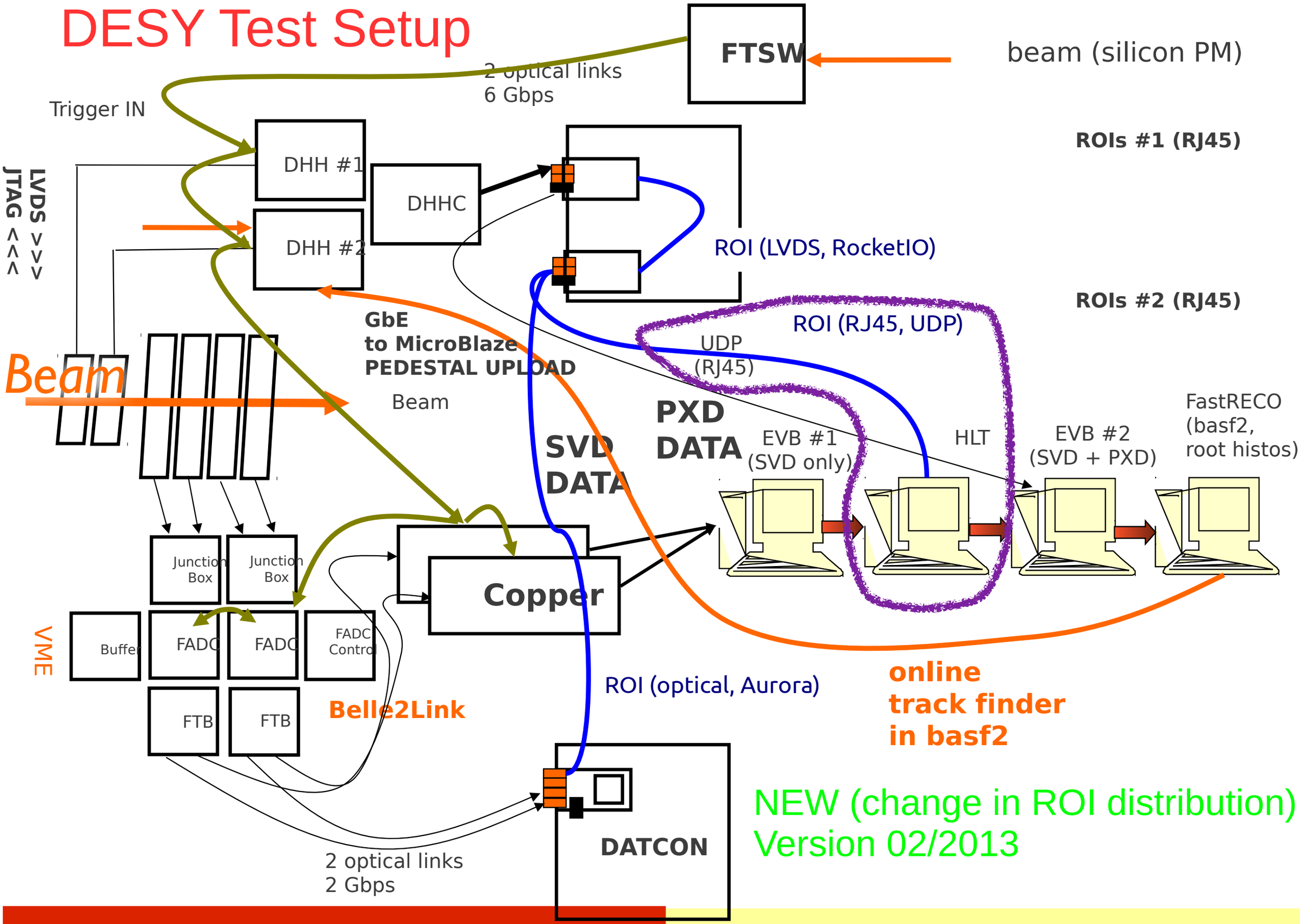
Conclusions

- PXDDataReduction Module is based on an pattern recognition module and a track-fit function whose performance influence the efficiency of the module
- The estimated ROI inefficiency of 20% (caveat emptor: pattern recognition inefficiency not included) is due failures in the fit and extrapolation of low transverse-momentum tracks
  - to increase the efficiency we should concentrate on the quality of the pattern recognition and of the fit of the track (genfit2 may help)
- The Data Reduction Factor is of the order of 1%, largely satisfactory
  - to be tested with background
- The performance of the module test-beam geometry are largely satisfactory from the points of view of efficiency, data reduction factor and execution time

Thank You!

backup-slides

DESY Test Setup



NEW (change in ROI distribution)
Version 02/2013

Parameters of the Study

Geometry:

```
geometry = register_module('Geometry')
param_geometry = {'Components': ['BeamPipe',
                                 'Cryostat',
                                 'HeavyMetalShield',
                                 'MagneticField',
                                 'PXD',
                                 'SVD',
                                 'SVD-Support',
                                 'CDC',
                                 ]}
geometry.param(param_geometry)
```

MC Track Finder:

```
mctrackfinder = register_module('MCTrackFinder')
mctrackfinder.logging.log_level = LogLevel.INFO
mctrackfinder.logging.debug_level = 101
param_mctrackfinder = {
    'UseCDCHits': 0,
    'UseSVDHits': 1,
    'UsePXDHits': 0,
    'MinimalNDF': 3,
    'UseClusters': 1,
    'WhichParticles': ['PXD', 'SVD'],
    'GFTrackCandidatesColName': 'mcTracks',
    }
mctrackfinder.param(param_mctrackfinder)
```

VXD Track Finder:

```
vxdtf = register_module('VXDTF')
param_vxdtf = {
    'GFTrackCandidatesColName': 'vxdtfTracks',
    'tccMinState': [2],
    'tccMinLayer': [3],
    'reserveHitsThreshold': [0.6],
    'sectorSetup': ['evtNormSecHIGH_SVD', 'evtNormSecMED_SVD', 'evtNormSecLOW_SVD'],
    'calcQIType': 'circleFit',
    'tuneCircleFit': [0.001, 0.001, 0.00001],
    'filterOverlappingTCs': 'hopfield',
    'cleanOverlappingSet': True,
    'activateZigZagXY': [False, True, True],
    'TESTERexpandedTestingRoutines': True,
    'qiSmear': False,
    }
vxdtf.param(param_vxdtf)
```

PXDDataReduction Module User Interface

maximum number of iteration of the kalman filter

σ_{syst} to be added in quadrature to σ_{stat} from the extrapolation

$N_{u,v}$, the total number of $\sigma = \mathrm{sqrt}(\sigma^2_{syst}+\sigma^2_{stat})$ to define the ROI

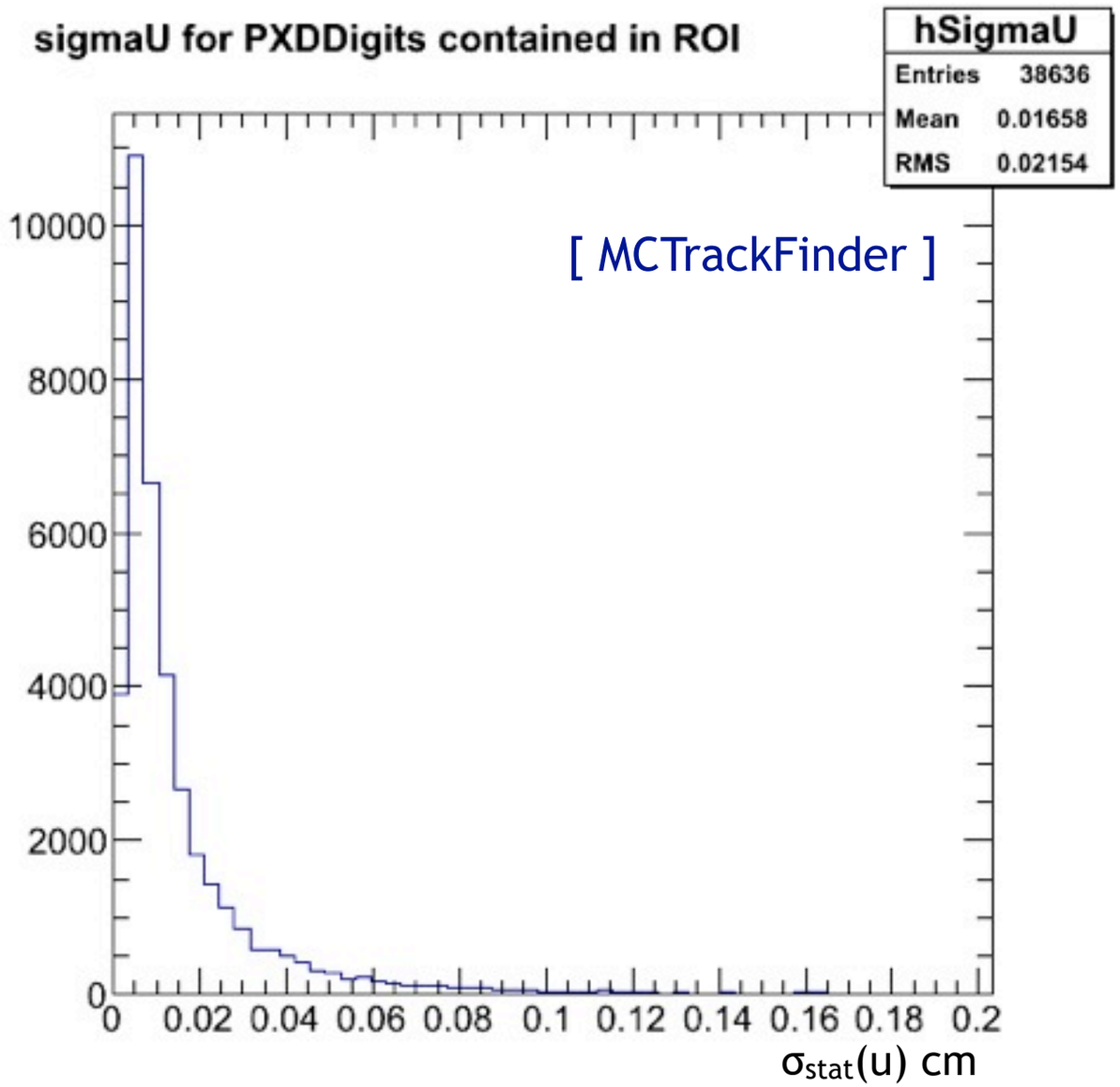
maximum allowed width of the ROI

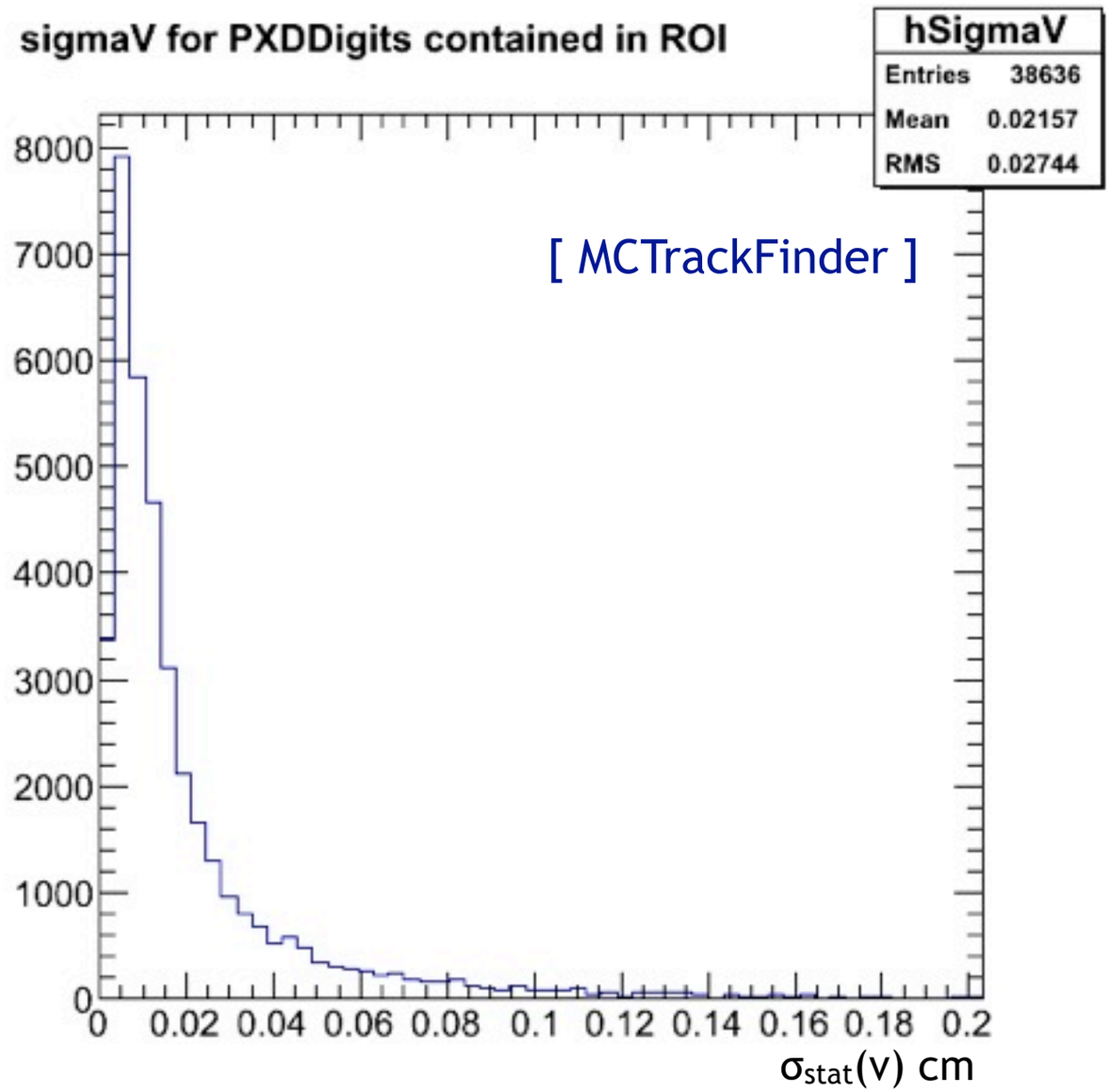
```
pxdDataRed = register_module('PXDDataReduction')
pxdDataRed.logging.log_level = LogLevel.INFO
param_pxdDataRed = {
    'trackCandCollName': 'mcTracks',
    'numIterKalmanFilter': 10,
    'PXDInterceptListName': 'PXDIntercepts',
    'ROIListName': 'ROIs',
    'sigmaSystU': 0.02,
    'sigmaSystV': 0.02,
    'numSigmaTotU': 10,
    'numSigmaTotV': 10,
    'maxWidthU': 0.5,
    'maxWidthV': 0.5,
    }
pxdDataRed.param(param_pxdDataRed)
```

$\min(\mathrm{maxWidthU}\ ,\ N_u * \mathrm{sqrt}[\ \sigma^2_{syst}(u) + \sigma^2_{stat}(u)\ ]\ )$

$\min(\ \mathrm{maxWidthV},\ N_v * \mathrm{sqrt}[\ \sigma^2_{syst}(v) + \sigma^2_{stat}(v)\ ]\ )$

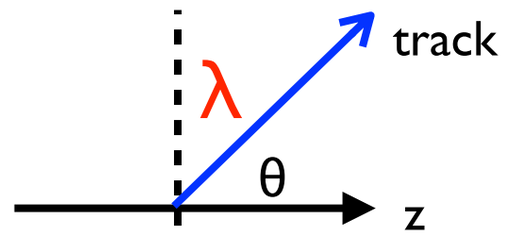
Statistical Error of the Extrapolation



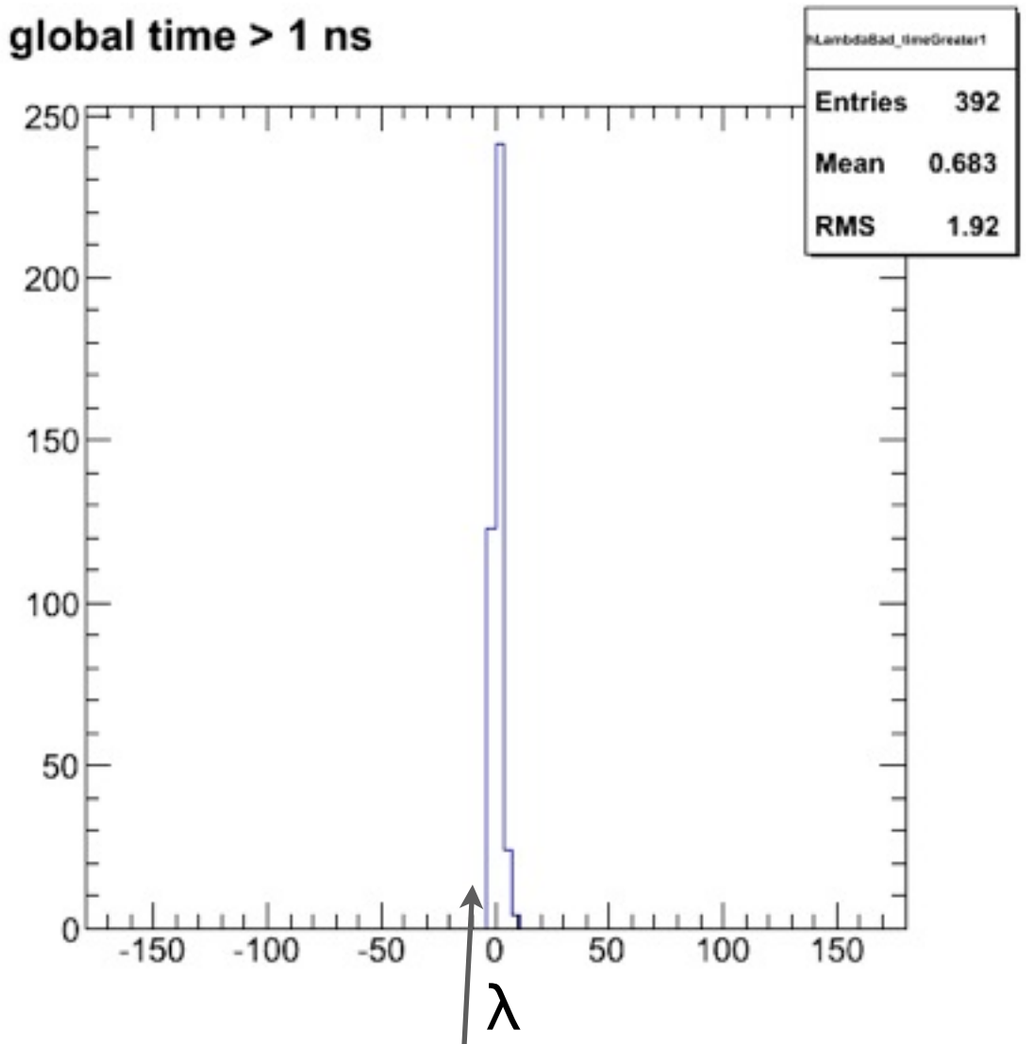


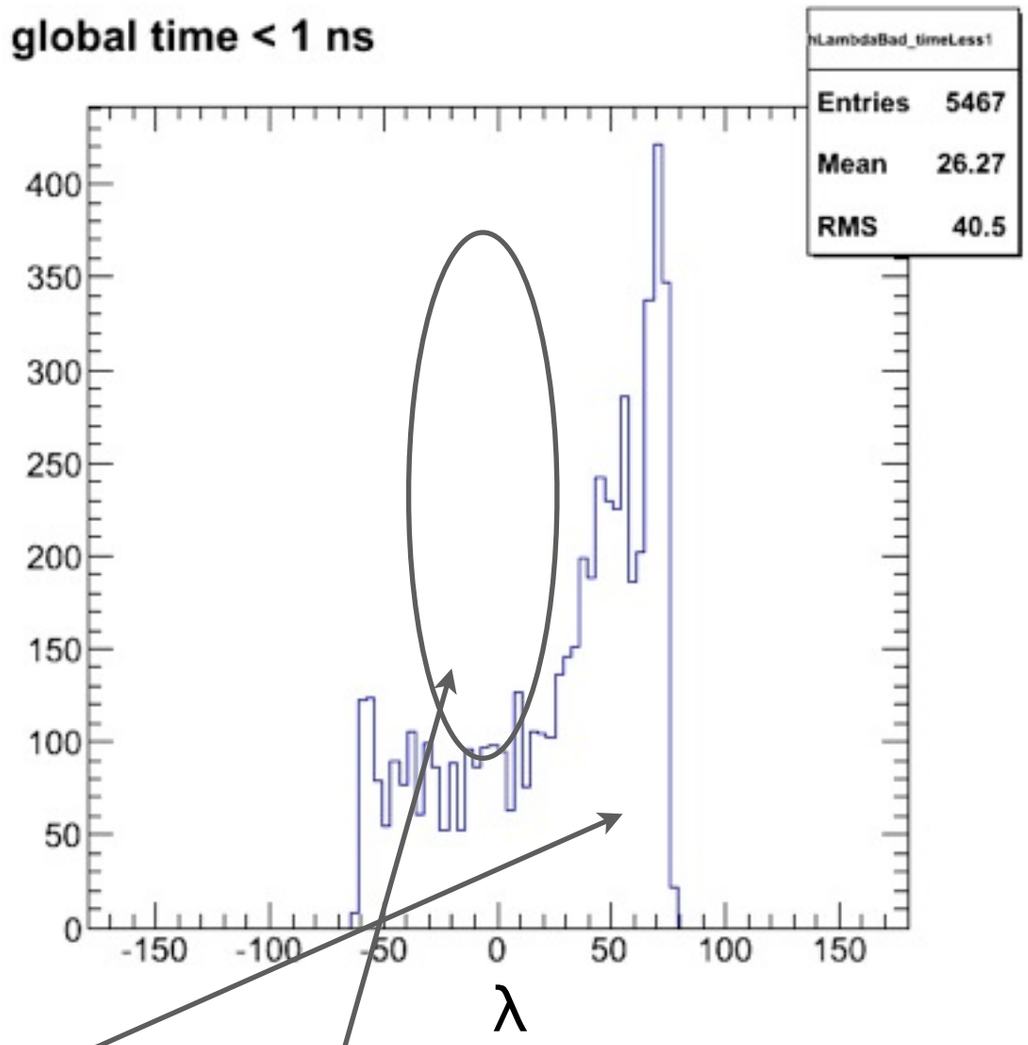
- with the VXDTF we observe similar statistical errors :
  - in the U direction: mean = 0.015 cm (-10%), RMS = 0.021 cm (+13%)
  - in the V direction: mean = 0.020 cm (-7%), RMS = 0.031 cm (-5%)

Main Source of Inefficiency



➡ tracks for which the fit has a bad track status or the intercept is not found:





➡ mainly hits of low transverse-momentum tracks

➡ later hits of tracks looping on the plane z ≈ 0 (7%)

➡ first hits of tracks λ≈65° (93%). First hits of track at λ=0 disappeared from the plot.